Micro-Grids: Practical Applications of Grid Technology to Small Distributed Collaborations

Jason Russell Lee

Lawrence Berkeley National Laboratory
Berkeley, CA,
JRLee@lbl.gov

Abstract. Recently there has been a great deal excitement about using both Computational and Data Grids [1,9], for large-scale scientific computing. However, the utility of Grid tools, ideas, and technologies to solve smaller-scale problems is often overlooked. In this paper, it will be shown not only that grids work very well for these smaller problems, but that the techniques developed for use in large scale grid computing can be applied equally as well to smaller problems and used in a effective and productive manner. This paper focuses on what Grid technology can be used to address a specific problem, particularly the aspects of monitoring a Grid and the associated problems and solutions. This will be described in the context of a scientific collaboration between six geographically distinct sites that spanned both the East and West coasts of the United States.

Introduction

In the research field of Micro-Electro-Mechanical Systems (MEMS), there are two major practical obstacles to experimental progress. The first is that fabrication and testing facilities are a scarce resource: only a handful of well-funded academic and government research programs have the equipment necessary to fabricate and perform experiments on-site, and commercial facilities are very expensive. The second obstacle is that scientists have difficulty sharing the results of their experiments with the rest of the community because the datasets are large, there is no common format for experimental results, and there is no uniform way of describing the experiment that produced the data. The solution to these problems is coordinated and transparent remote access to MEMS instruments and the resulting data from of MEMS experiments. This includes interactive control of test instruments, and a shared system for documenting experimental results so that they can later be retrieved with automated tools. The following sections will describe how Grid concepts, tools, and technologies were used to implement this solution in a DARPA-funded collaboration called Matisse [12] that involved (from West to East) the University of California at Berkeley (UCB), Lawrence Berkeley National Laboratory (LBNL),

Carnegie-Mellon University (CMU), the Information Science Institute (ISI-E), Sarnoff Laboratories, and Massachusetts Institute of Technology (MIT).

In this paper we show how Grid technology can be applied to help a small collaboration of researchers work together. We describe the distributed collaboration problem being solved, and describe how Grid data caches, monitoring systems, meta-data servers, and portals are used to facilitate the development cycle of MEMs devices. The primary aim of this work was to enable researchers to fully utilize all the available resources in a clear and simple manner. This was accomplished through the careful forecasting of available resources. This work is not complete by itself, and needs to be explained in the context of the entire Grid system that was used in the Matisse project.

MEMS Technology

Micro-Electro-Mechanical Systems (MEMS) are the integration of mechanical elements, sensors, actuators, and electronics on a common silicon substrate through the utilization of microfabrication technology. While the electronics are fabricated using integrated circuit (IC) process sequences (e.g., CMOS, Bipolar, or BICMOS processes), the micromechanical components are fabricated using compatible "micromachining" processes that selectively etch away parts of the silicon wafer or add new structural layers to form the mechanical and electromechanical devices.

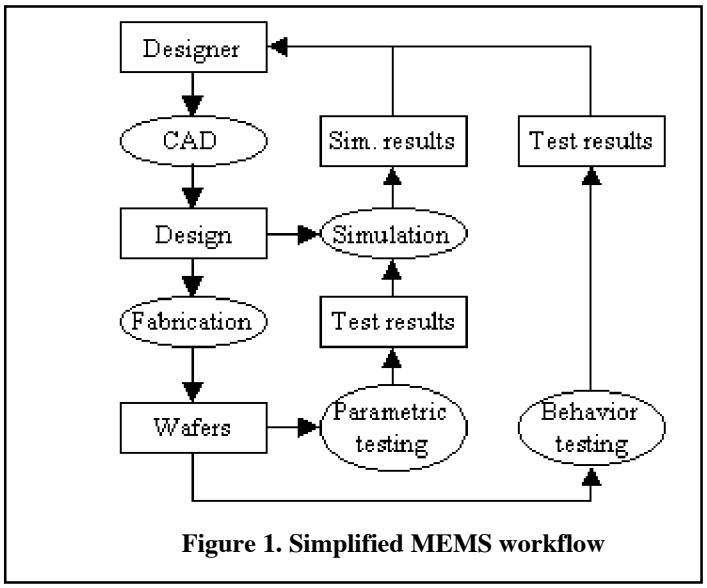

Figure 1. Simplified MEMS workflow

The workflow for creating MEMS devices, a simplified version of which is shown in

Figure 1, is complex. The basic idea is that the MEMS designer will generate a design in a Computer-Assisted Design (CAD) package, send that design to a fabrication site, then test the product (possibly comparing test results with simulation results), and use the images and computations from testing as feedback to the next round of designing. This process takes as input a design, test parameters, and test instrument settings, and produces as output large (many megabyte) files of test results.

Coordinating and storing the input and output datafiles, as well as providing remote access to and storing live images and instrument paramaters, is a time-consuming and error-prone task with simple tools such as remote login and FTP. Using grid technology to assist in this process greatly increases the productivity of the researcher. The particular aspect of grid technology that we are incorporating here is the ability to access and control all of resources associated with a project in a simple, but powerful manner. The end result of which will allow the researcher a powerful paradigm that provides a common interface to all the resources, thus allowing the designer to get real-time feedback during the fabrication of the devices. This architecture also allows for the exchange of designs, along with associated experimental results, among a larger community, thus creating a "virtual research space" for multiple institutions and researchers.

Matisse Project Resources

In the Matisse environment, all the resources were distributed across a wide geographical area. The six sites described above (UCB, LBNL, CMU, ISI-E, Sarnoff, and MIT) were heterogeneous in hardware, operating systems, and application software. Different sites contributed to the collaboration in different ways. Some sites had facilities for fabricating the chips for MEMS devices (UCB); some sites have testing facilities for the devices (UCB, MIT and CMU), and yet other sites (LBNL, ISI-E) have compute and storage resources. The sites and their respective facilities are shown in Figure 2.

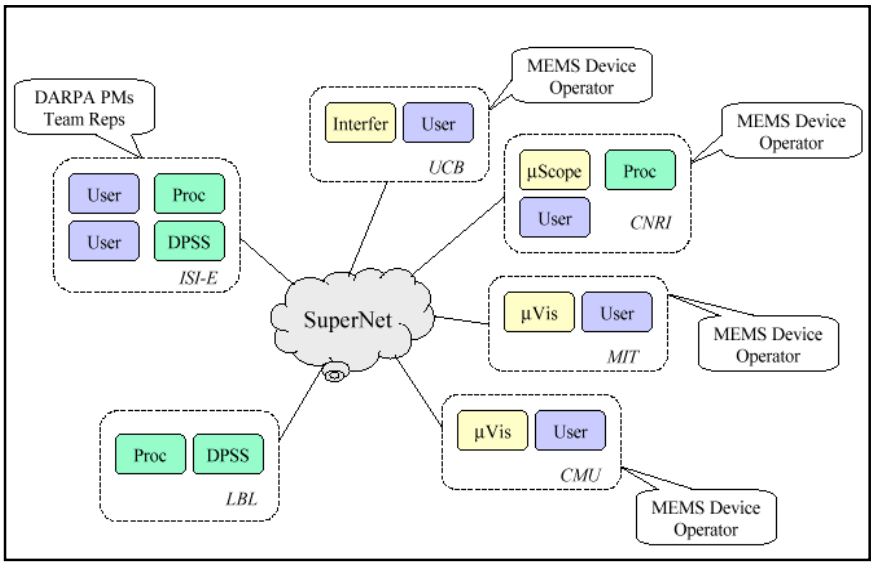

Figure 2 Matisse Grid

The collaborating sites were all connected by the SuperNet [2], a high-speed cross country network which is funded by DARPA[10]. The slowest network links in this environment are the single gigabit Ethernet connections into the end host machines. The Supernet cross-country trunk is an OC-48 (2.4 gigabits/sec), which is then sub-divided into single gigabit connections that connect into the various end sites. In addition to host computers, some of the MEMS testing devices such as interferometers were connected to the network.

The Matisse Grid

In order to allow transparent access to remote experiments, and to allow researchers to exchange their designs, the Matisse project used a data location and metadata service to both locate and add experiments to the system. A high-speed network cache held the large data files of streaming video and measurements. A Java-based portal client provided a unified metaphor for authenticating and accessing the system. Finally, real-time performance of the system was accessible through a monitoring service. These components are described in more detail in this section.

Data Location and Metadata Service

The Matisse system has a global data location and metadata service that is built on a CORBA[16] Object Request Broker (ORB) that interfaces to a back-end relational database. The interface allows registration of experiments, and all experiment-associated data, including CAD data, experimental runs, meta-data about the parameters used during fabrication. Much of the information stored is meta-data or handles, which in turn point to the real data locations through the use of URLs. This centralized name service scheme allows the designer the ability to login and check the status of the experiments from anywhere. It also provides for a dynamic method for the different fabrication sites to register the results of tests, and allows for changes during fabrication.

High-Speed Network Cache

This environment generates huge amounts of data, whether it is the 'movies' generated from the testing of a MEMS device or the volume data generated by the computation of the phase shift in a device during a run. High-speed access to this data is provided by the Distributed Parallel Storage System (DPSS)[8], a high-speed network cache. In the Matisse grid there are two DPSS's -- one on each coast of the United States -- to allow low-latency reading and writing of data from compute nodes and test facilities. A much more common method of transferring data, the File Transfer Protocol (FTP), was rejected because readily-available implementations of FTP have low (~8Mb/s) transfer rates on high bandwidth-delay product networks such as the SuperNet. The DPSS, on the other hand, can adjust its network performance based on the current network conditions, and has been shown to provide speeds in excess of 1.5 G/sec[17], which is well above the project's requirements.

Java Portal

Sarnoff designed a Java portal that provides integrated single sign-on access to the entire system. Users can login through this portal and check the status of their different projects. The portal provides the ability to share and compare the data from experiments with other researchers. A designer can take the data from one of their experiments and compare it against previous runs, or grant access to the data to other researchers who are doing similar work. Manufacturers can login to the system and see if there are new designs to waiting to be created, and the fabrication facilities have an easy way to now upload the data from the tests that they perform. One of the more important and key features of the portal is that it provides *transparent* access to all the resources through the use of a single sign-on. Once a user logs onto the portal, they now have access to all of

their resources through the portal. These resources can take the form of compute or storage resources, or the ability to upload new test data or designs to the system.

Grid Monitoring Service

In any network work environment it is vital to have information about the status of the network links. In a Grid environment it is imperative to also monitor all the resources that are attached to the network (i.e. the entire Grid), including physical resources such as free disk space and CPU load, and middleware resources such as storage servers and directory services. Then all the relevant information should be combined and used to make predictions about the future performance of the Grid. Being able to predict what is about to happen and answer questions such as; will a job finish shortly on the compute nodes, so they will be available? Will a network transfer finish shortly, so that the data from testing at a experiment can be streamed real-time back to the designer? These kinds of questions are very important and hard to answer, and need to be answered if one is to improve the performance of a 'grid' over that of simply a collection of loosely coupled machines. In the grid environment for the MEMS project we are using a program lblnettest[3] as a 'host-monitor" on each of the end hosts to provide a number of monitoring functions in the hopes of answering these questions.

1. End host conditions (cpu load, disk space avail, memory avail, etc)
2. Endwise network testing. (current network routes, latency, bandwidths)
3. Ability to add/trigger tests to obtain new data
4. Ability to monitor resources that can't run 'lblnettest' (switches,routers,etc)

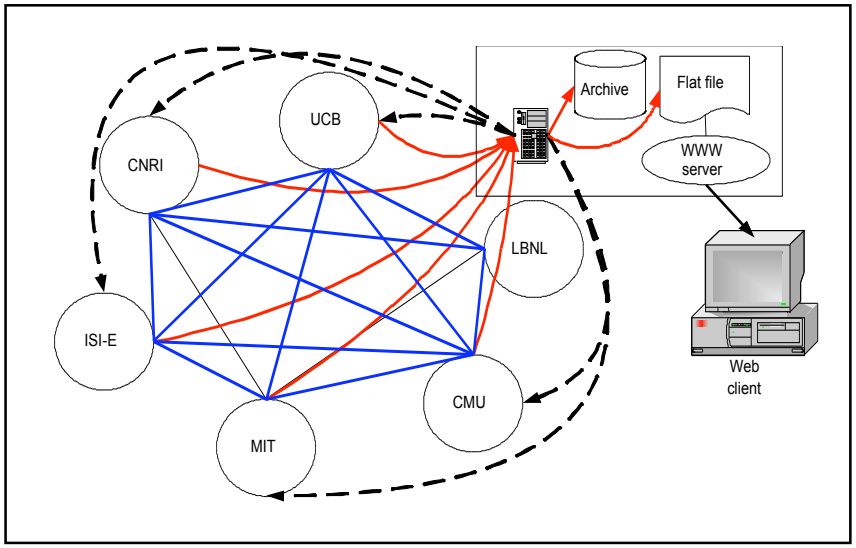

Figure 3. Matisse Monitoring

The program lblnettest provides a secure way to control sensors, allowing sensors to be manipulated and all their output to be captured and redirected. Lblnetest uses a public key infrastructure (PKI)[4] to control access to the sensors. In the Matisse grid environment we setup a public/private key pair assigned to a program, so that the program would have the ability to monitor all the resources on the grid.

lblnettest was able to perform continuous all-pairs (full mesh) testing of the network. This ensured that not only that connectivity was being maintained (using a udp ping equivalent), but also that an acceptable level bandwidth was available to the applications. The bandwidth testing was accomplished by doing measurements with a non-intrusive tool called "*pipechar*"[9] which is able measure the available (dynamic) bandwidth on a link without congesting the link. This technique, unlike the one used in *ttcp[13]*, *iperf[15]* and *nettest[3]* which inundate a link with traffic to discover the available bandwidth, sends out small number udp packets and measures the inter-arrival delay of the return packets to estimate the bandwidth. *lbltest* is also used to control the various host measuring tools such as vmstat, uptime, netstat or scripts that examine at the values in /proc (ie: /proc/loadavg on linux), and capture their output. This same method is utilized to execute programs that in turn monitor the status of the various resources that are connected to the grid.

Finally, all the output from the various sensors, both host and network alike, is parsed into a common logging format, called Universal Logging Format (ULM)[9] which is then sent via NetLogger [11] to a central logging server located at LBL. This logging server potentially now has enough information to make informed decisions about the future performance of the network and resources. And based on this information can recommended where computation jobs should be run and the data from these jobs stored.

Usage of the Matisse Grid

This section will provide an example of the normal use of the micro-grid we have been describing. A designer at MIT finishes the design of a new chip, and sends the design information off to the fabrication site located at UCB. UCB then starts the process of creating the chip, and at predefined intervals during the fabrication process, tests are run on the chip to verify that it stays within the design specifications. The data from each of these tests is stored locally, and then registered in a global namespace for later lookup by the designer. The designer at MIT can then peruse these tests at any time during the fabrication of the chip, and verify that they are within the design limits and possibly compare them to previous fabrication efforts to ensure that past mistakes aren't creeping in. This comparison process can take several different approaches: view the images from testing, view 'movies' of the testing of the device, or take all the data collected at the time of the test and run it through a compute farm extrapolating the phase, z-axis, deflection and any other fundamental data for the particular device being designed [7].

This last option is the one that the designer would most often like to perform, unfortunately it is computationally expensive and data intensive. However, with

transparent high-performance provided by the components described above, the designer does not need to optimize the process "by hand". Instead, the MEMS designer picks the tests of interest from a list gleaned from the global registry service of tests, selects analyses to perform, then presses the "*go*" button on the portal. This sets in motion several events, including verification that the necessary network, storage and computational resources are available for the desired analysis. After the analysis is complete, the data is either fed back to the designer, saved in a file for later review, or both.

The optimization that occurs under the hood will depend upon how the designer wants to see the results. If the data is fed back to the designer in real time, the entire process is optimized to start as early as possible, with continued progress until completion. With a small number of compute nodes and small number of high-speed caches on the network and the delay of the network either being very high or very low, this is a fairly straightforward optimization problem. The test data is transferred from the local test repository to the DPSS cache [9] (located closest to the greatest number of computational cycles). Then all the nodes on the network that have any amount of computation available (measured by not less then the RTT latency to the node from the cache) are utilized. The resulting data is streamed directly back to the DPSS cache that is closest to the designer, and from there onto the screen so that the results can be viewed.

If, instead, the results are stored for later review, the entire process is scheduled much the same as a batch processing job. The job is streamlined to store the end results back to persistent storage at the nearest cache to the computational nodes, and all computational cycles are considered for use with this run, as latency is no longer a factor.

If the designer wants to both store and review the results, then the optimizations must be made for real time viewing as in the first case, with the exception that the data is also streamed back to a network cache at the same time that it being viewed by the designer.

It should be apparent that at every juncture in the above process it is necessary to have information about the status of some resource. Questions such as; Is there enough space to store the results? Is there enough bandwidth for the user to view the results in real-time? Are there enough free cycles to do the analysis of X in Y amount of time? Not only need to be answered, but need to be answered in a timely fashion. Though the use of lblnettest we were successful at accomplishing this.

Future Work

The infrastructure for the Matisse project was successful in integrating diverse resources into a high-performance Grid, however we found that there were holes in the current set of tools available, and areas that we had not fully explored. One such area was the ability to show ‘correctness’ of previous runs/predictions by feeding previous data back into the analysis module and using the results to help predict the future with more accuracy.

Conclusions

In this paper we have shown some of the various problems facing Grid developers and some of the solutions that we have used to solve these problems. We have stressed the importance of monitoring all the resources in the network, which is in turn an important part of achieving *transparent* high-performance. Also necessary to good performance are high-speed caches that use monitoring information to 'tune' their behavior. Other ideas that we found to useful were the two modes of operation: near real-time, and a batch queue mode.

We have shown how Grid technology can be applied to many of today's problems, which appear to be more of an integration problem then a technology problem. Grids with transparent access show applicability towards a wide range of problems.

Acknowledgements

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division under U.S. Department of Energy Contract No,. DE-AC03-76F00098. This is report number LBNL-49216. See declaimer at http://www-library.lbl.gov/disclaimer.

References

1. "The Grid: Blueprint for a New Computing Infrastructure", edited by Ian Foster and Carl Kesselman. Morgan Kaufmann, Pub. August 1998. ISBN 1-55860-475-8.
2. SuperNet Network Testbed Projects: http://www.ngi-supernet.org/
3. Netest homepage http://www-itg.lbl.gov/nettest/download/download_info.html
4. Internet X.509 Public Key Infrastructure Certificate and CRL Profile (RFC 2459) http://www.ietf.org/rfc/rfc2459.txt
5. G. Jin, G. Yang, B. Crowley, D. Agarwal, "Network Characterization Service (NCS)", Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing HPDC-10, August 2001, LBNL-47892.
6. J. Abela, T. Debeaupuis. Universal Format for Logger Messages, IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-abela-ulm-05.txt
7. G. K. Fedder, S. Santhanam, M. L. Reed, S. C. Eagle, D. F. Guillou, M. S.-C. Lu, and L. R. Carley, "Laminated High-Aspect-Ratio Microstructures In A Conventional CMOS Process," Sensors & Actuators A, vol. A57, no. 2, pp. 103-110, March 1997.
8. Lee, Jason, "Design and Implementation of a Image Server System", Thesis submitted to San Francisco State University, Fall 1995, http://www-itg.lbl.gov/~jason/thesis/
9. The DataGrid Project: http://www.cern.ch/grid/
10. DARPA http://www.darpa.mil/
11. NetLogger Toolkit http://www-didc.lbl.gov/NetLogger/
12. The Matisse Project Homepage: http://www.mattise.org

13. TTCP src code: http://ftp.arl.mil/ftp/pub/ttcp/
14. NeTest home page: http://www-didc.lbl.gov/~jin/network/net-tools.html
15. Iperf home page: http://dast.nlanr.net/Projects/Iperf/
16. CORBA Reference Guide, The: Understanding the Common Object Request Broker Architecture, Alan Pope, Addison Wesley, Pub 1998 ISBN 0-201-63386-8
17. SC2000: http://www-didc.lbl.gov/presentations/SC00.LBNL.netchallenge.pdf